DESDM Workflow Discussion Meeting

Notes – Morning Session 13 February 2012

Intro – Brian Yanny

Looking for ideas from experience about simplifying and tuning up project workflow

Will keep discussion open, but practical considerations of budget and schedule will come in eventually

Basic unit of data taking is 1 camera exposure, size ~ 1GB; camera takes ~ 300 science exposures plus ~ 50 calibration exposures (same size) per night

Primer on calibration: take a number of order 10 for a calibration type (flat, sky flat, bias); average those to one image that is used to calibrate a full night (or longer) of data

Today's discussion will focus on the processing block called Single Epoch First Cut and on the blocks that pre-process the calibration data.

Single epoch pipeline hardware requirements:

- One 1-GB science image needs a node w/ 2-3 GB RAM, 20 GB scratch space
- 3-4 GB of calib images are needed per night
- CPU several (4-5) hours per exposure for full processing on one standard Linux box (shorter for simplified versions like supernova or community pipeline)
- Ingest to DB catalog is required at the end of exposure processing (not during? Question to be answered.)

Looking further at crosstalk module inside Single Epoch processing

Breaks a single exposure into 62 pieces

Generates a crosstalk matrix for further input to processing

Note: Filenames in system are NOT unique. Full path required to specify file. Would like to change this!

Note: The structure that passes flags into science codes does not fully/correctly capture provenance in part because the expansion occurs below the orchestration layer.

Single epoch processing requirement:

One night's data must be completed within 24 hours. Given CPU time above, implies need at least x300 nodes to meet this spec.

Data inflow: Exposures are sent from CTIO to NCSA throughout the night. Final image probably arrives 3-5 hours after end of data taking.

DESDM Components – Don Petravick

Build process – chaotic?

External packages are in ups

DESDM code is NOT!?

Not at all modular. Uses build 'sandbox', particular to an xsede node. Hard to have enough sandboxes (No kidding.)

No continuous build

GSN process IS the testing. Cycle is about a month.

Uses a file tree as a substitute for real unique file names. Transports subsets of tree to processing nodes, adds new files, copies back. Tree structure is part of wrapper (and code?)

Question: How are file permissions handled. Ans. Uses single unix group, all production users in same group.

Intermediate files not currently kept. Need to keep more, for restarts?

Copying across input files occurs at the beginning of a 'block', may occur between block steps if files are not available on that node.

Many DB interactions – see Don's slide for list. Would like to factor system to put DB heavy modules on NCSA machines – but constrained by policy?

Scripting and configuration

Orchestration reads .DES config files. These are not the complete configuration info.

Wrappers may be in either C or perl.

Science params are managed in svn.

Provenance captured in FITS header (NOT in DB?) and in file names/file tree. 1200 lines of perl in file_ingest. No calibration provenance in DB. Needed functionality (e.g. flag data processing with a questionable calibration file) is not possible (at least with finite effort) within the existing system. File_ingest takes 0.5 – 2 hours on a Lustre file system (used on xsede systems). File_ingest must be run even on failed processing in order to be able to retrieve files at all.

Normal automated running does not provide intermediate files. Debugging currently done with adding custom code. Failure analysis is done by diving into log files. Heavy human intervention is required to bypass errors.

DB is ORACLE – that will not change.

Converting to Python – can be done piecemeal.

NCSA DB can support production from Munich as well.

Brian Yanny – Next steps

Proposes to move toward a model which processes a single exposure rather than a single night

Requirements (high level)

- Preserve science results
- Must work for CP as well
- Must work in time for first data taking (how firm is that requirement, given current system works at some level?)
- Must be able to reproduce a processing run with same code and input data and params
- Must work on both xsede and osg style systems
- Must reduce effort required to debug and restart pipelines

Needed to get to requirements (high level)

- Add provenance model
- Reduce global file access bottleneck
- Add parameter management system, with versioning

- Add message logging system
- Uniform wrapper interface – disentangle DB/file access from codes?
- Restructure build process to allow clear versioning of code

More info needed

- Are there any *science* constraints preventing moving to single exposure model? This is answered by Michelle – no, just efficiency problems in current structure.
- Need full set of use cases for provenance
- Need complete picture of required DB access in current system, further DB access (if any) driven by provenance or error handling requirements

Afternoon Notes:

Existing File tree needs a redesign.

Need a way human-read file tree, while keeping low level names of

files unique.